

REMARKS/ARGUMENTS

Claims 1, 6-11, and 13-81 remain in the case.

Applicants gratefully acknowledge entry of the previous amendments to claims 19 and 39.

Claims 1, 6-11, 14-15, 77-78, and 81 remain rejected under 35 U.S.C. §102(e) as being anticipated by Forin (US 6,594,701). Claims 13, 16-27, 29, 31-45, 47-65, 67-76, 79 and 80 remain rejected under 35 U.S.C. §103(a) as being unpatentable over Forin in view of Dunning et al. (US 6,683,850). Claims 8, 28, 30, 46 and 66 remain rejected under 35 U.S.C. §103(a) as being unpatentable over Forin in view of Dunning et al. (US 6,683,850) further in view of Cheriton et al. (US 6,724,721).

Applicants initially note that there is disagreement over the definition of the word “range”, the Examiner taking the position that “range” can be a single attribute scalar quantity that does not describe the extent of another parameter, while Applicants insist that a “range” requires a representation that encompasses a plurality of attributes, such as begin and end, begin and quantity, etc., that describes an extent of another parameter. In fact, resolution of this issue may not be required, as the claims do not simply use the word “range”, but rather qualify the word range with other language, which is discussed below.

For example, claim 1 employs the phrase, “defining a ... unique range of data”, and indeed, the claims discuss two such defined “unique range[s] of data”. If we assume *arguendo* that the word “range” can mean “amount” or “quantity”, as asserted by the Examiner, we are left with an inconsistency, as to how a single scalar value could itself define something “unique”. In fact, Forin uses the quantity of credits to **infer** the range as being an adjacent sequential block, but this inference is different than a **defined** unique range, which does not rely on any such inference. Therefore, even if we assume *arguendo* that the examiner’s interpretation of the word “range” alone is plausible, the remaining analysis is inconsistent with the words “defined” and “unique”, each of which is attributed with no meaning in accordance with the Examiner’s interpretation. It is axiomatic in claim interpretation that all claim language be given distinct meaning, and therefore the Examiner’s interpretation is flawed. It is noted that proper construction of claims is not limited to the courtroom, and seeking such interpretation does not comprise a “new issue requiring further consideration.”

In fact, the meaning of the phrase “define a [first/second] unique range of data” is clearly that the information may not be abstracted to the level of a single scalar value, and must include

sufficient information to **define** a **unique range** of data, rather than merely permit the recipient to infer the intended range of data from a single value, and strictly limit that inference *a priori*. In the later case, there is no flexibility within the protocol, and for example, any breach of the protocol requires a separate communication, as is expressly disclosed by Forin.

Indeed, a careful review of Forin indicates that information defining a unique range of data is indeed available, and further that information is employed in generating the “credits”, which are alone communicated without the underlying unique range information. The system in accordance with Forin intentionally filters the information sufficient to define a unique range, to specify only a set of “credits” which are absent such range information. Forin acknowledges as well that, because information has been lost in the filtering, and because communications rely on various presumptions, the result may be a protocol failure, the solution to which is a reestablishment of the communications and communication of the range information using a separate scheme. In contrast, the present invention provides a system and method in which the actual unique range of data is defined, and therefore no such inference is necessary or warranted. If a transmission of data is corrupted, the specific data to be retransmitted may be identified and can be retransmitted without exiting the protocol or producing a fault or requiring the entire set. Because Forin provides a system and method in which the only inference is that credits represent sequential portions, the protocol provides no way to specify any other range but the inferred next sequential one. It is respectfully submitted that the system and method according to Forin therefore fail to **define** a **unique range** of data, as required by the claims, but rather simply infer a unique range of data, based on a hopefully shared presumption by the sender and receiver.

Forin indeed do address this problem, and provide that the information necessary to initialize the presumptions is sent separately in information packets as “descriptors”. These descriptors, however, do not meet the present claim limitations, nor has the Examiner asserted that they do.

In contrast to the analysis by the Examiner, this distinction is expressly made in the language of the independent claims, and is not inferred only by reference to the specification.

Forin states (emphasis added):

In order to control the flow of data without copying the data, the receiver may communicate application-level **receive buffer sizes** to the sender. The receiver preferably communicates the buffer size information to the sender in an efficient manner. For example, the more buffer size information communicated to the sender in each flow control communication, the more efficient the communication process. In one implementation, the receiver may communicate a list containing at least one application-level receive buffer size to the sender, so that the sender can determine how much data the receiver is capable of

receiving. In preferred implementations of the invention, the receiver may send a list containing a plurality of application-level receive buffer sizes to the sender. One method for communicating the list of buffer sizes to the sender is by sending a message, e.g., a packet, from the receiver to the sender over a data channel established between the sender and the receiver. The message may contain the list of receive buffer sizes, and is hereinafter referred to as a credit message. The receive buffer sizes in the credit message are hereinafter referred to as credits.

The sender may utilize the credits in the credit message to determine the size and order of data packets to be sent to the receiver. For example, the sender preferably does not exceed the size indicated by a particular credit or send data when no credits are available. In addition, the sender preferably uses the credits in the order that the credits are received from the receiver, so that the receiver can receive data into the correct buffers. Because the credits are preferably indicative of application-level receive buffer sizes, the data sent by the sender can be received directly into allocated application-level receive buffers. Thus, the credit-based flow control methods and systems according to the invention provide both reliable and efficient data transfer between senders and receivers.

Since credits may be received in a finite-sized buffer managed by the sender, the flow of credits from the receiver to the sender is preferably controlled. In order to control credit flow, the receiver may utilize the receipt of data from the sender as an indication that there is a buffer available to receive new credits. For example, the sender preferably only sends data to the receiver when the sender has been notified through a credit list that a receive buffer is available. Thus, when the receiver receives data from the sender, the receiver knows that a previous credit list has been successfully communicated to the sender. When the sender receives new credits from the receiver, the sender preferably posts a new receive buffer to receive additional credits. The sender is preferably prevented from using credits in the new credit list until the buffer for receiving the next credit list is posted. Thus, when the receiver receives data from the sender corresponding to the first credit in a new credit list, the receiver also knows that a buffer for receiving additional credit lists is available. **One additional assumption made by the receiver** is that the sender initially, i.e., before any credit messages or data is transferred, has at least one buffer available for receiving credit lists. Finally, the size of the credit list is preferably no greater than the size of the sender's credit list buffer or the network MTU between the sender and receiver, whichever is smaller. Thus, based on these rules, the present invention reliably implements flow control of credits.

According to another aspect, the present invention includes a method for controlling data flow between a sender and a receiver. The method includes communicating a first credit list to a sender. The first credit list may include a plurality of credits indicative of buffer sizes of receive buffers accessible by the receiver and capable of receiving data from the sender. In response to receiving the first credit list, the sender transmits a data packet to the receiver. The data packet is no greater in size than a first buffer size specified by a first credit in the first credit list.

According to another aspect, the present invention may include a network communications system. The network communication system may include a first local virtual interface, a second local virtual interface, and a credit list builder/communicator. The first local virtual interface may send data to and receive data from a first remote virtual interface over a first network connection. The second local virtual interface may send credit messages to and receive credit messages from a second remote virtual interface over a second network connection. The credit list builder/communicator may build credit messages for controlling data flow over the first network connection and communicate the credit messages to the second remote virtual interface through the second local virtual interface and the second network connection. The credit messages may include credit lists including a plurality of credits indicative of buffer sizes of receive buffers for receiving data through the first local virtual interface from the first remote virtual interface. Alternatively, each virtual interface may be used to communicate data in one direction while communicating credit messages in the reverse direction.

According to an important aspect of the invention, the receiver 62 communicates credits to the sender 60 to control the flow of data packets 84 sent by the sender 60. In the illustrated embodiment, communicating the credits to the sender includes sending credit messages 82 to the sender. The credit messages may be variously configured. In a preferred embodiment, the credit messages may include credit lists containing buffer size information relating to the size of one or more receive buffers 78 in which the receiving application 74 may request receipt of data. In order to generate credit lists and communicate credits to the sender, the receiver may include a credit list builder/communicator 83. When the receiving application requests receipt of data into one or more of the receive buffers 78, e.g., by communicating the buffer virtual addresses and sizes to the I/O device interface 80, the credit list builder/communicator 83 may generate credit messages including the sizes of the receive buffers 78 and forward the credit messages to the I/O device 76 to be sent to the sender 60. Alternatively, the credit list builder/communicator may communicate credits to the sender using a shared memory buffer or through RDMA write operations, as previously described. The credit list builder/communicator 83 may also determine when to communicate new credits to the sender 60. Methods for determining when to communicate new credits to the sender 60 are discussed in more detail below.

According to an important aspect of the invention, the credit list reader/processor 75 preferably uses the credits to determine the size of data packets to be sent to the receiver. In addition, the credit list reader/processor preferably uses the credits in the order that the credits were received, so that the receiver will receive data in the correct buffers. For example, the credit message 82 may indicate that the receiving application 74 has a first buffer of four bytes for receiving data and a second buffer of two bytes for receiving data. The sending application 66 may have a send buffer of six bytes to be sent to the receiving application. **Under these conditions, the credit list reader/processor 75 may request that the I/O device 70 send a first data packet of four bytes and a second data packet of two bytes to the receiver 62, e.g., by communicating the virtual addresses of the data to be sent along with the appropriate sizes to the I/O device 70.** The credit list reader/processor 75 preferably maintains a list of credits received from the receiver 62 and removes credits from the list as the sender uses the credits. Thus, because the receiver 62 preferably communicates credits indicative of application buffer sizes to the sender, and the sender 60 constructs data packets having sizes based on the credits, data flow between the sender and the receiver may be efficiently regulated. Moreover, software copying, segmentation, and reassembly of data may not be required according to preferred implementations of the invention because the data packets sent to the receiver are preferably no greater in size than the corresponding receive buffers.

FIG. 3 is a more detailed block diagram of the sender 60 and the receiver 62 illustrated in FIG. 2. The sender 60 and the receiver 62 illustrated in FIG. 3 preferably implement the Virtual Interface Architecture (VIA). According to the VIA architecture, the efficiency of I/O operations may be increased by granting I/O devices direct access to application-level data buffers so that copying of data between applications and the I/O devices is not required. In order to provide I/O devices direct access to application-level buffers, the I/O device interfaces 72 and 80 communicate descriptors to the I/O devices. **A descriptor is a data structure containing I/O request processing information, such as the virtual memory address and size of a send or receive buffer. The I/O devices translate the virtual memory addresses in the descriptors to physical memory addresses and either send data from or receive data into a buffer at the physical memory address. The buffer size information in the descriptors may also be used by the credit list builder/communicator 83 to generate credit messages.**

The virtual interfaces 96 and 97 may comprise communication interfaces between the sending and receiving applications 66 and 74 and the I/O devices 70 and 76. The virtual interface 96 of the sender 60 may include a send queue 98 and a receive queue 99. Similarly, the virtual interface 97 of the receiver 62 may include a send queue 100 and a receive queue 101. In order to request an I/O operation, the sending and receiving applications 66 and 74 may execute standard I/O commands, such as Winsock send() and Winsock recv(). In response to these commands, the VI user agents 89 and 91 may post descriptors 102-109 to the send and receive queues of the sender and the receiver and notify the I/O devices of the posting of the descriptors. Posting the descriptors may include writing pointers to the virtual memory addresses of the descriptors to the virtual memory addresses of the queues. Notifying the I/O devices of the descriptors may include writing doorbell tokens including the virtual memory addresses of the descriptors to virtual memory addresses of doorbells associated with each queue. The VI kernel agents may map the virtual memory addresses of the doorbells to physical memory addresses of doorbell control registers associated with the I/O devices. When the I/O devices 70 and 76 receive doorbell tokens, the devices preferably increment a descriptor counter for the associated queue. The I/O devices 70 and 76 decrement the counters when descriptors are processed. The I/O devices 70 and 76 preferably process the descriptors in the order that the descriptors are posted in the send and receive queues and perform the requested I/O operations. The I/O devices preferably process the descriptors until the queues are empty or until an unrecoverable error occurs.

In order to request the sending of data, the sending application 66 may transmit a request for sending data to the VI user agent 89, which posts descriptors 102 and 103 in the send queue 98 of the sender virtual interface 96 and rings the doorbell of the send queue 98 once for each descriptor. The descriptors 102 and 103 may specify the virtual memory addresses of send buffers 68 containing data to be sent to the receiver. Once the descriptors are posted in the send queue 98, the I/O device 70 of the sender preferably locates the data at the virtual memory addresses indicated in the descriptors and sends the data to the receiver.

In order to receive data from the sender, the receiving application 74 preferably sends receive data requests to the VI user agent 91, which posts descriptors 106 and 107 in the receive queue 101 specifying one or more receive buffers 98 to store data from the sender. However, if no descriptors are posted in the receive queue 101 when the data arrives, connection between the sender and receiver may be broken. Similarly, because the receiver may not perform segmentation or reassembly of data, if data in a given data packet from the sender exceeds the size of the receive buffer in the descriptor 107 at the head of the receive queue 101, connection may also be broken. Accordingly, it is desirable to coordinate posting of descriptors in the send queue 98 of the sender with the posting of descriptors in the receive queue 101 of the receiver; i.e., it is desirable to control flow between the sender and the receiver.

In order to control flow between the sender 60 and the receiver 62, the credit list builder/communicator 83 builds credit messages based on sizes of the receive buffers contained in receive data requests initiated by the receiving application 74. For

example, when the receiving application posts a descriptor in the receive queue, the credit list builder/communicator 83 may record the size of the buffer specified by the descriptor in a credit message. The credit list builder/communicator 83 may repeat this process for each descriptor posted in the receive queue. When the number of credits in the receive queue reaches a predetermined value or when the credit list builder/communicator 83 determines that the sender needs credits, the credit list builder/communicator 83 preferably requests that the I/O device 76 send a credit message 82 to the sender. Methods for determining when the sender needs credits will be discussed in more detail below.

Credit messages may be sent to the sender in any suitable manner, for example, by posting a descriptor in the send queue 100 of the receiver containing the size and virtual memory address of the credit message and ringing the send queue doorbell. However, in a preferred embodiment of the present invention, the sender and the receiver use a separate connection from the connection(s) for sending and receiving data for the sending and receiving of credits. ... **Each credit or credit message may specify the data connection or virtual interface to which it pertains.** ...

Once a credit message is transmitted to the sender, the credit list reader/processor 75 of the sender receives the credit message 82. Receiving a credit message may require the previous posting of a descriptor containing the virtual address and size of a credit message buffer in the receive queue 99 of the sender. Alternatively, as discussed above, credits may be received over a separate connection from the connection for receiving data. The credit list reader/processor 75 may use the credits in the credit message to control the posting of descriptors in the send queue 98. For example, the sender may have a send buffer containing six bytes of data to be sent to the receiver. The credit message 82 from the receiver may contain a first credit of four bytes and a second credit of two bytes, indicating that the descriptors 106 and 107 specify two-byte and four-byte receive buffers, respectively. Accordingly, the credit list reader/processor 75 may post a first descriptor 103 in the send queue 98 containing a pointer pointing to the first byte of the send buffer with a size of four bytes and a second descriptor 102 in the send queue 98 containing a pointer pointing to the fifth byte of the send buffer 68 with a size of two bytes. The I/O device 70 may process the descriptor 103 and transmit a first data packet having four bytes of data to the receiver. The I/O device 70 may process the second descriptor 102 and transmit a second data packet of two bytes of data to the receiver. When the receiver receives the data packets, the receiver processes the descriptor 107, then the descriptor 106, to store the received data packets in four- and two-byte receive buffers, respectively. In this manner, the sender only sends data that the receiver is capable of receiving. As a result, data transmission overflow errors are reduced and transmission efficiency is increased.

* * *

In order to regulate the flow of data from the sender, the credit list builder/communicator 83a of the receiver may generate a list of credits based on descriptors posted in the receive queue 101. The list of credits may be stored in memory of the I/O device 76 or in memory of the host computer in which the I/O device is inserted. The credit list builder/communicator 83a may send the credit list to the sender by instructing the I/O device 76 to send the list directly from the memory location in which the credit list is stored. The credit list builder/communicator 83a may also determine when to communicate new credits to the sender as will be discussed in more detail below.

The credit list reader/processor 75a may receive the credit list and process the credits in order to send data to the receiver. However, unlike the credit list reader/processor 75 illustrated in FIG. 3, rather than posting descriptors in the send queue, the credit list reader/processor 75a may control the sending of data specified by descriptors previously posted in the send queue 98 of the sender so that the size of data packets actually sent to the receiver corresponds to the credits. For example, a descriptor specifying the sending of eleven bytes of data may be located at the head of the send queue 98. The credit list reader/processor 75a may have two credits of five bytes and six bytes. Accordingly, the credit list reader/processor 75a may break the data buffer specified by the descriptor into a first data packet of five bytes and a second data packet of six bytes. Thus, when the credit list reader/processor and the credit list builder/communicator are implemented in hardware, flow control can be achieved transparently to the VI user agents 89 and 91.

As stated above, the credit list builder/communicator 83 preferably determines when to communicate new credits to the sender. Determining when to provide the sender with new credits may be accomplished in any number of ways. FIG. 4 illustrates exemplary steps which may be performed by the credit list builder/communicator 83 to determine when to communicate new credits to the sender. In step ST1, the credit list builder/communicator 83 may receive requests for receiving data from the receiving application 74. **The credit list builder/communicator 83 preferably determines the size of the receive buffer in each request and adds a credit of a corresponding size to a credit list.** Step ST1 is preferably executed repeatedly and concurrently with the remaining steps in FIG. 4 to accumulate credits as requests are received from the receiving application 74. In steps ST2 and ST3, the credit list builder/communicator 83 determines whether the number of accumulated credits exceeds a predetermined number or whether the sender has no credits. The predetermined number of credits may be based on a maximum credit message length, which may be determined by the smaller of the network MTU between the sender and the receiver and the size of the buffer posted by the sender to receive credit messages. If either condition is satisfied, the credit list builder/communicator 83 may communicate a first batch or list of credits to the sender. (ST4) For example, the credit list builder/communicator may instruct the I/O device 76 to send a first credit message to the sender, e.g., by posting a descriptor having a pointer to the credit message in the send queue of the control connection of the receiver and ringing the send queue

doorbell. In an alternative embodiment, for example, where the sender and receiver communicate using shared memory, the credit list builder/communicator 83 may write the credit list to the control portion of the shared memory if both conditions are satisfied. If neither of the conditions is satisfied, the credit list builder/communicator may continue to accumulate credits. In steps ST5 and ST6, the credit list builder/communicator 83 determines whether data has been received from the sender for the first buffer specified in the first batch of credits communicated to the sender. If data has not been received in the first buffer, the credit list builder/communicator 83 preferably continues checking whether data has been received in the first buffer, i.e., without communicating a new list of credits to the sender. If the credit list builder/communicator 83 determines that data has been received in the first buffer specified in the first credit list, the credit list builder/communicator 83 determines whether new credits are available. (steps ST7 and ST8) If new credits are available, the credit list builder/communicator 83 preferably communicates a new credit list to the sender. (step ST9) For example, the credit list builder/communicator may instruct the I/O device 76 to send a new credit list to the sender containing newly accumulated credits. The newly accumulated credits may be based on receive buffers contained in data receive requests initiated by the receiving application after the previous credit message was sent. If there are no newly accumulated credits, the credit list builder/communicator 83 may continue to check until new credits are available. After the new credit list is communicated to the sender, the credit list builder/communicator 83 determines whether data has been received in the first buffer specified in the new credit list. (steps ST10 and ST11) If data has not been received in the first buffer in the new credit list, the credit list builder/communicator 83 preferably continues checking, i.e., without communicating another new credit list to the sender. If the credit list builder/communicator 83 determines that data has been received in the first buffer in the new credit list, the credit list builder/communicator 83 preferably checks whether new credits are available and instructs the I/O device 76 to send another new credit list to the sender.

Claims 19, 39 and 57 employ the term “specified” or “specifying” a range rather than “defining” as is used by claim 1, which likewise distinguishes an inferred range.

Claim 19 provides, *inter alia*:

“(i) transmitting credits from a receiving node to a sending node responsive to occurrence of an event, said credits specifying a unique range of data to be transmitted;
(ii) transmitting a **specified** unique range of data of a data stream from said sending node to said receiving node, corresponding to a range of data **specified** in credits received by said sending node from said receiving node”....

Claim 39 provides, *inter alia*:

“a) means for transmitting a range of data of a data stream from a sending node to a receiving node, said range of data being **specified** by a range of data credits present at said sending node;
b) means for transmitting a number of data credits **specifying** the range of data of said data stream from said receiving node to said sending node upon occurrence of at least one event”....

Claim 57 provides, *inter alia*:

“b) a first transmitter for transmitting an amount of data of a data stream from a sending node to a receiving node, corresponding to a range of data specified by credits present at said sending node, if said predetermined identifier indicates implementation of a credit and negative acknowledgement transport system;
c) a second transmitter for transmitting credits from said receiving node to said sending node when a predetermined event occurs, said credits **specifying** a range of data sought to be received; and for transmitting a negative acknowledgement from said receiving node to said sending node, when at least one transmitted datum is lost or corrupted.”

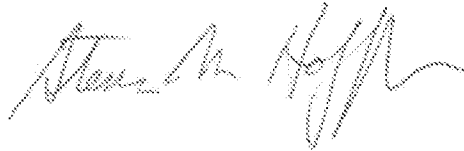
In particular response to the Examiner, the Examiner states in the advisory action: “This

also means that this scalar quantity implies that a group of data extending over the range of this actual number will be sent (i.e. the limitation “defining a first unique range of data”), which is an inherent feature when one specifies a number of bytes to send.” In fact, the Examiner admits that the credit merely “implies” the range of data, but does not define or specify it. Likewise, while in context, the begin and end locations may be “inherent”, it is only because of the range I itself defined or specified by the “descriptor” of Forin, and not because of information conveyed in the credit message itself. Since Forin uses a separate communication to actually define or specify the range, it must be concluded that the credit message does not provide this information, and therefore does not meet the present claim language.

It is therefore respectfully submitted that the Forin reference is deficient in at least one material element of the claims, and does not anticipate, nor in combination with the secondary references, render obvious, the claims.

Applicants respectfully request reconsideration, submit that claims 1, 6 - 11, and 13 - 81 are allowable.

Respectfully submitted,

A handwritten signature in dark ink, appearing to read 'Steven M. Hoffberg', written in a cursive style.

Steven M. Hoffberg
Reg. 33,511

Milde & Hoffberg LLP
Suite 460, 10 Bank Street
White Plains, NY 10606
(914) 949-3100